leetcode Count of Range Sum

2016年1月10日 (https://www.hrwhisper.me/leetcode-count-of-range-sum/) hrwhisper (https://www.hrwhisper.me/author/hrsay/) 16 Comments (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comments) 7,143 views

> **leetcode Count of Range Sum**
>
> Given an integer array `nums`, return the number of range sums that lie in `[lower, upper]` inclusive.
>
> Range sum `S(i, j)` is defined as the sum of the elements in `nums` between indices `i` and `j` (`i` ≤ `j`), inclusive.
>
> **Note:**
>
> A naive algorithm of *O*(n^2) is trivial. You MUST do better than that.
>
> **Example:**
>
> Given *nums* = `[-2, 5, -1]`, *lower* = `-2`, *upper* = `2`,
>
> Return `3`.
>
> The three ranges are : `[0, 0]`, `[2, 2]`, `[0, 2]` and their respective sums are: `-2, -1, 2`.

题目地址： leetcode Count of Range Sum (https://leetcode.com/problems/count-of-range-sum/)

题意:

给定一个整数组成的数组，求它的所有子区间和坐落于[lower, upper] 的个数。

比如样例中[-2, 5, -1]中，这三个区间的和在[-2,2]之间 [0, 0], [2, 2], [0, 2]

思路

先来看看最朴素的O(n^2)算法，首先算出和，然后枚举区间范围。

```java
public class Solution {
    public int countRangeSum(int[] nums, int lower, int upper) {
        if(nums.length == 0) return 0;
        long[] sum = new long[nums.length + 1];
        for (int i = 0; i < nums.length; i++)
            sum[i + 1] = sum[i] + nums[i];

        int ans = 0;
        for (int i = 0; i < nums.length; i++) {
            for (int j = i + 1; j <= nums.length; j++) {
                if(lower  <= sum[j] - sum[i] && sum[j] - sum[i] <= upper)
                    ans++;
            }
        }
        return ans;
    }
}
```

题目要求的效率好于O(n^2)的算法，那么要怎么加速呢?

还记得 Count of Smaller Numbers After Self (https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/) 么?

那时候，我们用Fenwick树或者线段树，先离散化，然后从右向左扫描，每扫描一个数，对小于它的求和。然后更新.....

这题也差不多，需要找满足条件 lower ≤ sum[j] – sum[i – 1] ≤ upper， 也就是 lower + sum[i – 1] ≤ sum[j] ≤ upper + sum[i – 1]

我们同样的求出和，然后离散化，接着从右向左扫描，对每个i 查询满足在 [lower + sum[i – 1], upper + sum[i – 1]]范围内的个数（用线段树或者Fenwick Tree）

这样复杂度就是O(n log n)

线段树

C++

```cpp
typedef long long LL;
struct SegmentTreeNode {
    LL L, R;
    int cnt;
    SegmentTreeNode *left, *right;
    SegmentTreeNode(LL L, LL R) :L(L), R(R), cnt(0), left(NULL), right(NULL)
};

class SegmentTree {
    SegmentTreeNode * root;
    SegmentTreeNode * buildTree(vector<LL> &nums, int L, int R) {
        if (L > R) return NULL;
        SegmentTreeNode * root = new SegmentTreeNode(nums[L], nums[R]);
        if (L == R) return root;
        int mid = (L + R) >> 1;
        root->left = buildTree(nums, L, mid);
        root->right = buildTree(nums, mid + 1, R);
        return root;
    }

    void update(SegmentTreeNode * root, LL val) {
        if (root && root->L <= val &&  val <= root->R) {
            root->cnt++;
            update(root->left, val);
            update(root->right, val);
        }
    }

    int sum(SegmentTreeNode * root, LL L, LL R) {
        if (!root || root->R < L ||  R < root->L ) return 0;
        if (L <= root->L  && root->R <= R) return root->cnt;
        return sum(root->left, L, R) + sum(root->right, L, R);
    }

```

```cpp
public:
    SegmentTree(vector<LL> &nums, int L, int R) { root = buildTree(nums, L, R

    int sum(LL L, LL R) {
        return sum(root, L, R);
    }

    void update(LL val) {
        update(root, val);
    }
};

class Solution {
public:
    int countRangeSum(vector<int>& nums, int lower, int upper) {
        if (nums.size() == 0) return 0;
        vector<LL> sum_array (nums.size(),0);
        sum_array[0] = nums[0];
        for (int i = 1; i < sum_array.size(); i++) {
            sum_array[i] = nums[i] + sum_array[i - 1];
        }
        LL sum = sum_array[sum_array.size() - 1];
        sort(sum_array.begin(), sum_array.end());
        auto t = unique(sum_array.begin(), sum_array.end());
        SegmentTree tree(sum_array, 0, t - sum_array.begin() - 1);
        int ans = 0;
        for (int i = nums.size() - 1; i >= 0; i--) {
            tree.update(sum);
            sum -= nums[i];
            ans += tree.sum(lower + sum,upper + sum);
        }
        return ans;
    }
};
```

Binary indexed tree (Fenwick tree)

关于此树的介绍： Binary indexed tree (Fenwick tree)
(https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/)

注意要加入upper 和 lower -1 两个点

(python 版本比C++简洁太多了^ ^,建议看py版本)

C++

```cpp
typedef long long LL;
class FenwickTree {
    vector<int> sum_array;
    int n;
    inline int lowbit(int x) {
        return x & -x;
    }

public:
    FenwickTree(int n) :n(n), sum_array(n + 1, 0) {}

    void add(int x, int val) {
        while (x <= n) {
            sum_array[x] += val;
            x += lowbit(x);
        }
    }

    int sum(int x) {
        int res = 0;
        while (x > 0) {
            res += sum_array[x];
            x -= lowbit(x);
        }
        return res;
    }
};

class Solution {
```

```cpp
public:
    int countRangeSum(vector<int>& nums, int lower, int upper) {
        if (nums.size() == 0) return 0;
        vector<LL> sum_array (nums.size() * 3,0);
        LL sum = 0;
        for (int i = 0; i < nums.size(); i++) {
            sum += nums[i];
            sum_array[i * 3] = sum;
            sum_array[i * 3 + 1] = sum + lower - 1;
            sum_array[i * 3 + 2] = sum + upper;
        }
        sum_array.push_back(upper);
        sum_array.push_back(lower - 1);
        unordered_map<LL, int> index;
        sort(sum_array.begin(), sum_array.end());
        auto end = unique(sum_array.begin(), sum_array.end());
        auto it = sum_array.begin();
        for (int i = 1; it != end;i++,it++) {
            index[*it] = i;
        }
        FenwickTree tree(index.size());
        int ans = 0;
        for (int i = nums.size() - 1; i >= 0; i--) {
            tree.add(index[sum],1);
            sum -= nums[i];
            ans += tree.sum(index[upper + sum]) - tree.sum(index[lower + sum -
        }
        return ans;
    }
};
```

Python

```python
class FenwickTree(object):
    def __init__(self, n):
        self.sum_array = [0] * (n + 1)
        self.n = n

    def lowbit(self, x):
        return x & -x

    def add(self, x, val):
        while x <= self.n:
            self.sum_array[x] += val
            x += self.lowbit(x)

    def sum(self, x):
        res = 0
        while x > 0:
            res += self.sum_array[x]
            x -= self.lowbit(x)
        return res


class Solution(object):
    def countRangeSum(self, nums, lower, upper):
        """
        :type nums: List[int]
        :type lower: int
        :type upper: int
        :rtype: int
        """
        if not nums: return 0
        sum_array = [upper, lower - 1]
        total = 0
        for num in nums:
            total += num
            sum_array += [total, total + lower - 1, total + upper]

        index = {}
        for i, x in enumerate(sorted(set(sum_array))):
            index[x] = i + 1

        tree = FenwickTree(len(index))
        ans = 0
        for i in xrange(len(nums) - 1, -1, -1):
            tree.add(index[total], 1)
            total -= nums[i]
```

```
                ans += tree.sum(index[upper + total]) - tree.sum(index[lower + tot
            return ans
```

本文是 leectode 327 Count of Range Sum 的题解，

更多leetcode题解见 https://www.hrwhisper.me/leetcode-algorithm-solution/ (https://www.hrwhisper.me/leetcode-algorithm-solution/)

分享到：　新浪微博　微信　QQ空间　人人　腾讯微博　豆瓣　印象笔记

Leetcode (https://www.hrwhisper.me/category/code/leetcode/) c++ (https://www.hrwhisper.me/tag/c/), Data Structure (https://www.hrwhisper.me/tag/data-structure/), leetcode (https://www.hrwhisper.me/tag/leetcode/), python (https://www.hrwhisper.me/tag/python/), 算法 (https://www.hrwhisper.me/tag/algorithm/). permalink (https://www.hrwhisper.me/leetcode-count-of-range-sum/).

leetcode Power of Three (https://www.hrwhisper.me/leetcode-power-of-three/)

二叉树最近公共祖先详解（LCA问题详解） (https://www.hrwhisper.me/algorithm-lowest-common-ancestor-of-a-binary-tree/)

16 thoughts on “leetcode Count of Range Sum”

szh says:
2016年5月24日 at pm8:49 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1406)

请问为什么是lower – 1而不是lower?

Reply (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1406#respond)

hrwhisper (https://www.hrwhisper.me) says:
2016年5月24日 at pm9:10 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1408)

给你个提示 就是 树状数组求区间和是进行相减的

ly (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1408#respond)

szh says:
2016年5月24日 at pm9:15 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1409)

但是我觉得把lower – 1都换成lower对整体上好像没影响，求解。。

s://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1409#respond)

hrwhisper (https://www.hrwhisper.me) says:
2016年5月24日 at pm9:22 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1411)

都换成lower就变成了 lower + sum[i – 1] < sum[j] ≤ upper + sum[i – 1] 本来第一个是 lower + sum[i – 1] ≤ sum[j] 的

w.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1411#respond)

szh says:
2016年5月24日 at pm9:33 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1413)

啊，多谢！请问LZ这些数据结构和算法是在哪里学的？需要专门买书看吗？

hrwhisper (https://www.hrwhisper.me) says:
2016年5月24日 at pm10:50 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1415)

以前看ACM的时候学的 刘汝佳《算法竞赛入门经典训练指南》

zzhzzh says:
2016年5月16日 at am9:10 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1307)

求问楼主为什么从右往左扫啊?

Reply (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1307#respond)

hrwhisper (https://www.hrwhisper.me) says:
2016年5月24日 at pm9:18 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1410)

lower ≤ sum[j] – sum[i – 1] ≤ upper 要求j > i

ly (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1410#respond)

zzhzzh says:
2016年5月24日 at pm9:27 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1412)

哦哦哦，原来是这样，感谢博主，看了好多BIT的解法都没看懂，就你这个实现和讲解最直观了

ps://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1412#respond)

hrwhisper (https://www.hrwhisper.me) says:
2016年5月24日 at pm10:49 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-1414)

不客气^ ^

w.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=1414#respond)

xtq says:
2016年4月11日 at pm5:38 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-792)

有个问题，可能是我对树状数组理解不深，就是如果是数组中间的一部分在区间里面，这个是怎么计算进去的，因为我看每次total都是减去末尾的一个元素，然后重复下去，这样子每次add和sum不都是算从头到当前位置那部分的吗？谢谢。

Reply (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=792#respond)

hrwhisper (https://www.hrwhisper.me) says:
2016年4月17日 at pm5:09 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-817)

抱歉回复晚了. 树状数组就是每次计算的从头到当前位置的和。 你可以看博客中关于树状数组的介绍 （本文就有链接啦）

ply (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=817#respond)

xtq says:
2016年4月18日 at am5:10 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-819)

恩。。。那到底是怎么考虑进去数组中间的一部分和到底符不符合要求呢。。。我理解能力差。。。看到网上就你写的关于这道题比较全面，麻烦你的回答了~

:ps://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=819#respond)

hrwhisper (https://www.hrwhisper.me) says:
2016年4月19日 at am8:51 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-826)

数组中间是否满足要求？ 不是求 lower ≤ sum[j] – sum[i – 1] ≤ upper => lower + sum[i – 1] ≤ sum[j] ≤ upper + sum[i – 1] 么？ 每次对区间进行查询啊 在区间的就是符合要求的。

ww.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=826#respond)

zwl says:
2016年1月11日 at am12:26 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-244)

题主做hard类型的题目大概要多久。 hard题目相当于acm什么难度的题目。 谢谢。

Reply (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=244#respond)

hrwhisper (http://www.hrwhisper.me/) says:
2016年1月11日 at pm12:52 (https://www.hrwhisper.me/leetcode-count-of-range-sum/#comment-246)

看情况，有思路就比较快= = 以这题来说 应该算中等吧。。

ply (https://www.hrwhisper.me/leetcode-count-of-range-sum/?replytocom=246#respond)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment